오델로 인공지능 설계 보고서

1. 어떤 전략을 사용했는가?

* 가장 기본 적으로 위의 사진에 적힌 기준으로 가중치를 설정하여, 오델로가 놓을 자리를 계산을 하였다. 가장 우선적인 부분은 **각 4개의 모서리**이며, 그 이후 에는 **파란색의 가중치가 높게 설정**이 되어있다. 하지만 그림에 나오는 것은 일반적인 가중치 일뿐 코드에서는 조금 다르게 설정을 하였다.
* 사진에 보면 다음과 같이 **1/90**으로 설정한 가중치는 쉽게 말하여 내가 해당 자리에 놓을 때 **바로 옆인 모서리가 나의 돌일 경우 가중치를 90**으로 주고, **만약 아무것도 없거나, 다른 돌일 경우에는 가중치를 1**로 주어 가장자리의 모서리를 먹지 못하게 하였다.
* 또한 파란색 라인에서는 내가 놓으려고 하는 곳에 **좌우가 나와 같은 색이거나, 아무것도 없을 경우에는 해당 색상에 보이는 가중치를 주**고, 만약 내가 놓을 자리에 **좌, 우 중 한곳이라도 다른 색 돌이 있을 경우에는 가중치를 1**로 주게 만들었다.
* 해당 방법을 제외하고도, 처음 시작에서부터 10번 정도 턴이 지나갈 때 까지는 위의 가중치를 사용하지 않고, 내가 **현재 놓을 돌과 상대방이 놓을 수 있는 가장 최고의 자리를 계산하여, 그 중 상대방이 가장 적게 뒤집히는 위치**로 돌을 놓게 하였다.
* 10번의 턴이 지난 후 에는, **상대방의 돌이 나의 돌보다 많거나, 상대방이 2개이상의 모서리를 가지고 있거나, 내가 3개이상의 모서리를 가지고 있을 경우 에는 가중치를 우선**적으로 사용하게 하였고, 정반대로 **나의 돌이 많을 경우에는 상대방이 가장 적게 뒤집히는 자리**를 선택하여 돌을 놓게 하였다.

1. 어떻게 프로그래밍 했는가?

* 가장 기본적으로 각각의 Black, White 모두 배열을 가지고 있는다. 배열의 초기화는 Black의 경우 처음 Attack 시 “bool isBlackStart\_2012180004 = false” 변수를 만들어 false일경우 배열을 초기화하여 가지게 하였고, White의 경우 Defence 시 bool 값을 판단하여 배열을 초기화 하였다.
* 배열은 “isArr, isAiNext” 두 개의 배열을 만들게 되는데, 하나의 배열은 시스템 판의 배열과 동일하게 진행을 하면서 가중치를 구하기 위한 힌트를 구하거나 할 때 사용을 하며, 두 번째 배열은 한 수 앞을 보기 위한 배열로 사용된다.
* 한 수 앞을 보는 방법은 내가 놓을 수 있는 모든 자리를 놓음과 동시에 상대방의 놓을 수 있는 최상의 자리를 놓는 것을 모두 계산하여, 그 중 가장 나의 돌이 적게 넘어가는 것을 고르는 방식이다. 처음 시작 시 임시 배열을 하나 만들어서 해당 배열에 현재 진행중인 판을 저장하고, 자리를 놓고선 해당 기록을 저장 후 임시 배열에 있는 것을 다시 원래 배열에 넣는 방식으로 모든 자리를 체크하게 만든다.
* 우선 가중치를 두는 것은 아주 간단한 프로그래밍 이였다. 위의 사진에서의 좌표를 자세히 살펴서, 해당 좌표에 해당 돌이 놓을 수 있으며, 그 좌표의 돌에 스코어 점수를 준다. 돌을 놓을 수 있는 자리를 계산을 하는 것은 “void isNextHintLoad\_2012180004()” 라는 함수에서 돌의 놓을 수 있는 자리를 불러온다. 다 불러오게 되면 저장되는 데이터는 “x, y, 뒤집을 수 있는 돌의 개수” 를 최대 20개까지 배열에 저장을 하며, 저장된 배열을 바탕으로 void isAiTotalPlus(int whatPlay, int scoreData[]) 함수에서 지정된 가중치를 뒤집을 수 있는 돌의 개수에 더하기를 시킨다.
* 가중치를 두는 방법에서도 몇몇의 경우에는 상황에 따라 변화는 가중치가 있는데, 이러한 가중치들은 if문을 통하여 해당 위치에 특정한 돌이 있거나, 없는 경우에 따라서 점수를 다르게 준다.
* 처음 가중치를 줄 때에는 “isArr, isAiNext” 각각의 배열에 가중치를 각각 주어야 해서, 함수를 두 개를 만들어 하나를 수정하면, 뒤에 하나도 따라서 수정을 하게 되어 있었는데. 프로그래밍을 하다 보니 작성을 두 번 하는 것이 너무 힘들어서, 위의 함수와 같이 어떤 배열인지와, 해당 가중치를 더해줄 스코어 부분을 배열로 받아서 하나의 함수로 작성을 할 수 있게 수정을 하였다.

1. 배틀 수행 결과 보고서

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 구분 | 박원준 | 조승환 | 김은재 | 김하늘 | 원준영 | 이정석 |
| 권창현 | 1승 3패 | 2승 2패 | 0승 4패 | 4승(실격승) | 4승(실격승) | 4승(실격승) |
| 최종 결과 | | | | | | |
| 15승 9패 | | | | | | |

1. 오델로 Term프로젝트 수행 소감

* AI를 처음 제작하게 된 것은 2013년 4월 2일에 내준 5번문제인 “각각의 순서에 따라 오델로 판에 돌을 놓을 수 있는 모든 자리를 표시하라.” 라는 문제를 풀고 난 후 AI에 대해서 생각을 하게 되었다. 처음 AI를 생각할 때에는 무조건 많이 뒤집으면 돌이 많아지니 이기기 싶겠다라는 생각으로 힌트를 배열에 담아서 해당 자리 중 가장 많이 뒤집어 지는 곳에 좌표를 두게 만들었다. 처음에는 내가 직접 돌을 놓는 게 아닌 컴퓨터가 알아서 둔다는 것에 정말 기분이 좋았는데. 몇 번 대결을 하고 나니깐, 컴퓨터가 너무 허무하게 지니깐 좀 더 생각을 잘하게 하는 AI를 만들어 보자 하면서 제작을 하였다. 그렇게 하여 제작된 것이, 가중치를 두는 부분과, 나의 돌이 많이 넘길 수 있게 한 수 앞을 보는 등의 여러 가지 방법의 코딩을 하였다.
* 오델로 AI를 실제로 대전을 하고 난 후 생각할 겨를도 없이 가장 좋은 것은 랜덤으로 아무 위치에다가 놓는 것이 가장 좋은 AI라고 생각을 하였다. 물론 랜덤이 무조건 좋은 것이 아닌, 잘 짜인 AI일 경우에는 랜덤보다는 생각을 하는 AI가 훨씬 좋은 것은 당연한 것이다. 하지만 우리가 생각할 수 있는 프로그래밍 수준과 실력에서는 아무리 고민을 하고 생각을 해서 놓는 AI라 해도 한계가 드러나는 것은 당연할 수 밖에 없다. 이럴 경우에는 심각한 고민 보다는 랜덤 한 위치와 조금의 가중치를 생각하면 오히려 더 많은 게임에서 승리를 거두었을 것이라는 생각이 든다.
* 마지막으로 AI를 제작하면서 인공지능이란 게 얼마나 어려운지 깨닫게 되었다. 수정을 하면 할수록 오히려 안 좋아지거나, 좀더 좋게 수정을 하려면 내가 그게임을 완벽하게 파악을 하고 그 게임에서의 실력이 어느 정도 이상이어야지만 인공지능을 짤 수 있다는 것을 깨닫게 되었다. 끝으로 C로만들고 system(“cls”)를 하다 보니 너무 깜빡임이 심한 것을 느껴서 요번 학기에 배운 윈도우프로그래밍을 바탕으로 오델로를 좀더 실제 게임과 같은 프로그램을 제작해보려고 한다. 물론 완전히 새로 짜는 것이 아닌 기존의 C로 만든 코드를 복 붙을 하면서… 그래픽만 추가하는 방식이 될 것 같다..!